NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

AN APPROACH FOR DETECTING MALICIOUS EMAILS USING RUNTIME MONITORING WITH HIDDEN DATA

by

Kristin R. Sellers

September 2016

Thesis Advisor:	Doron Drusinsky
Second Reader:	Man-Tak Shing

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** September 2016	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE AN APPROACH FOR DETECTING MALICIOUS EMAILS USING RUNTIME MONITORING WITH HIDDEN DATA	**5. FUNDING NUMBERS** HDTRA 139119
6. AUTHOR(S) Kristin R. Sellers	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) Defense Threat Reduction Agency (DTRA), 8725 John J. Kingman Rd., Fort Belvoir, VA 22060	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol number ____N/A____.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.	**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

Computer systems continue to be at risk of attack by malicious software that are attached to email. Email has been determined to be the cause of 80% of computer virus infections. Millions of dollars are lost yearly due to the damage brought by malicious emails. Popular approaches toward the defense against malicious emails are antivirus scanners and server-based filters. Further, state-of-the-art methods are being employed to enhance security against malicious programs. However, despite efforts being subjected toward the protection of personal information in emails, malicious programs continue to pose a significant threat.

This thesis presents the application of a hybrid of Runtime Monitoring and Machine Learning for monitoring patterns of malicious emails. The system is designed in a way that it gathers malicious emails to determine whether they are suspicious, unknown, or benign. The application of runtime monitoring helps reduce the chance that suspicious emails are spread and lowers the likelihood that users will be threatened. Patterns were developed in Rules4business.com to facilitate the detection of threats and apply rules to the identified rules validation, while at the same time tracking them. The runtime monitoring application system entails the detection of the malicious emails by assessing the pattern in which they are sent and qualifying them into different states identified as suspicious, unknown, or benign. Through the application of the system, it would be possible to eliminate threats posed to private individuals and corporations emanating from the malicious emails.

We performed deterministic runtime monitoring, built a Hidden Markov Model (HMM), and performed runtime monitoring with hidden data. It is the reasoning about the patterns of malicious emails with hidden artifacts that provides the potential of providing improved classification.

14. SUBJECT TERMS malicious emails, runtime monitoring, statechart assertions, formal specifications, Hidden Markov Model	**15. NUMBER OF PAGES** 59
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

AN APPROACH FOR DETECTING MALICIOUS EMAILS USING RUNTIME MONITORING WITH HIDDEN DATA

Kristin R. Sellers
Lieutenant, United States Navy
B.S., Langston University, 2008

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
September 2016

Approved by:	Dr. Doron Drusinsky Thesis Advisor
	Dr. Man-Tak Shing Second Reader
	Dr. Peter Denning Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Computer systems continue to be at risk of attack by malicious software that are attached to email. Email has been determined to be the cause of 80% of computer virus infections. Millions of dollars are lost yearly due to the damage brought by malicious emails. Popular approaches toward the defense against malicious emails are antivirus scanners and server-based filters. Further, state-of-the-art methods are being employed to enhance security against malicious programs. However, despite efforts being subjected toward the protection of personal information in emails, malicious programs continue to pose a significant threat.

This thesis presents the application of a hybrid of Runtime Monitoring and Machine Learning for monitoring patterns of malicious emails. The system is designed in a way that it gathers malicious emails to determine whether they are suspicious, unknown, or benign. The application of runtime monitoring helps reduce the chance that suspicious emails are spread and lowers the likelihood that users will be threatened. Patterns were developed in Rules4business.com to facilitate the detection of threats and apply rules to the identified rules validation, while at the same time tracking them. The runtime monitoring application system entails the detection of the malicious emails by assessing the pattern in which they are sent and qualifying them into different states identified as suspicious, unknown, or benign. Through the application of the system, it would be possible to eliminate threats posed to private individuals and corporations emanating from the malicious emails.

We performed deterministic runtime monitoring, built a Hidden Markov Model (HMM), and performed runtime monitoring with hidden data. It is the reasoning about the patterns of malicious emails with hidden artifacts that provides the potential of providing improved classification.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

CSV	Comma Separated Values
DOD	Department of Defense
HMM	Hidden Markov Model
IP	Internet Protocol
IRS	Internal Revenue Service
LTL	Linear-Time Temporal Logic
MTL	Metric Temporal Logic
NL	Natural Language
R4B	Rules4Business (web service)
REM	Runtime Execution Monitoring
RM	Runtime Monitoring
RV	Runtime Verification
SME	Subject Matter Expert
UML	Unified Modeling Language

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I pass my gratitude to my academic advisors, Dr. Doron Drusinsky, my thesis advisor; Dr. Man-Tak Shing, second reader; Dr. Peter Denning, chair at the Department of Computer Science; and my writing coach, Michelle Pagnani. I am grateful for their constant support during the entire period of study.

I further extend my gratitude to my family members and friends for the endless support during the whole period.

This research was funded by a grant from the U.S. Defense Threat Reduction Agency (DTRA).

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

™Email has some time for now been an internet executioner application used by people, organizations, and governments for imparting, sharing and dispersing data. However, a range of illegitimate emails is among the emails sent out. Certain fraudulent actors, for example, those connected with spam use email to send spontaneous mass ads to influence people to buy items that will create income. Other actors, for instance, those behind phishing use email as a means to obtain an individual's biodata and to profile people who are susceptible to these types of activities. The analysis and monitoring of various types of malicious emails are focused on in the thesis.

The thesis concentrates on analyzing temporal and sequencing patterns of malicious emails using both visible email data as well as learned hidden state information; it then used a hybrid run-time monitoring technique to qualify suspicious email sequences.

Based off information in the emails, we developed three categories for the hidden states: suspicious, unknown, and benign. For example, if an individual is constantly receiving an email from a fraudulent actor, we would identify the pattern and classify the hidden state as suspicious. We will use these three states as inputs to the runtime monitoring algorithm described in the sequel.

An assertion, or rule, is a mathematical rule used to predict behavior. In software engineering, "assertion is a statement that a predicate (Boolean-valued function, a true-false expression) is expected to always be true" [1]. The formal specification assertion can monitor the sequencing and the temporal patterns of the malicious emails. By categorizing the emails using assertions, we are also able to compare the behavioral patterns to the correct behavior as specified by a formal specification [2].

The approach taken in this thesis is as follows. First, we developed deterministic rules to detect threats based on temporal and sequencing patterns; by deterministic it is meant that the rule assumes all its inputs are visible (have a 0 or 1 probability of occurrence). We then validated those rules by applying them to the known threats. Next, we generated the Hidden Markov Model using a machine learning technique. Finally, in runtime, we used the validated rules to input data that contains both visible and hidden artifacts, for detection and tracking of incoming threats.

Our input email-data is packaged as Microsoft Excel worksheets. Variations of these csv files were used to (i) perform deterministic runtime monitoring for rule validation, (ii) helped build deterministic rules for monitoring hidden and visible data, (iii) build and generate a Hidden Markov Model (HMM) in the learning phase, and finally (iv) to perform runtime monitoring with hidden data.

A. THE NEED FOR RUNTIME MONITORING OF MALICIOUS EMAILS

Often computer security threats encompass execution of unauthorized foreign code on the victim machine [3]. Malicious emails received with links or attachments serves as security threats are one example of unauthorized code. In Fiskiran and Lee's paper [3], "Runtime Execution Monitoring (REM) to Detect and Prevent Malicious Code Execution," they say "REM can detect program flow anomalies that occur during execution such as buffer overrun attacks commonly used by network and malicious emails." They conclude by asserting the need for formal methods to effectively categorize malicious emails.

This thesis uses a runtime monitoring program to present formal specifications as a way to detect malicious emails and to distinguish the hidden artifacts in an email. Runtime monitoring provides real-time situational awareness of conditions, a quality mentioned in the Fiskiran and Lee's paper [3]. In addition, by using temporal assertions, we demonstrate the detection of sequential patterns of emails. Temporal assertions detect patterns of emails that users may not evident from a single email. Therefore, sequencing and temporal patterns of emails is potentially more informative than monitoring individual emails one by one, independently of each other. This topic is further addressed again in Chapter III.

B. MOTIVATION FOR USING RUNTIME MONITORING OF HIDDEN DATA

Every day users are receiving massive amount of emails. With intruders seeking information or hiding their intent by mimicking well-known websites, the user may ask themselves, “can I trust this email?” A straightforward answer is “analyze the content of the email” (i.e., analyze each email independently of others). This answer, however, fails to exploit sequencing and temporal information associated with a plurality of emails. Hence, an improved approach, demonstrated in this thesis, is to monitor sequences and temporal patterns of emails. Monitoring sequences of emails is potentially more informative than monitoring individual emails because it helps distinguish a hidden intent of the email sequence, an intent that is not evident from individual emails.

For example, suppose we receive an email from an agent that works for the IRS and uses the same format as the IRS. The agent states that the organization has identified cases of fake agents sending out emails and asking for personal information, but in the content of this email, the agent also asks for contact information. Within the next two days, we receive an email from a different agent, but this individual is also using the same domain. This time, the agent requests date of birth. Receiving both emails within a week, the sequence if more suspicious than each individual email alone.

An additional contribution of this thesis is that it demonstrates monitoring sequences of emails where some email properties are not contained in the email text (i.e., they are hidden properties). These properties are probabilistically learned and modeled as a Hidden Markov Model (HMM). Runtime monitoring of temporal and sequencing patterns of emails based on both visible and hidden artifacts has the potential to provide even better discovery of malicious email patterns.

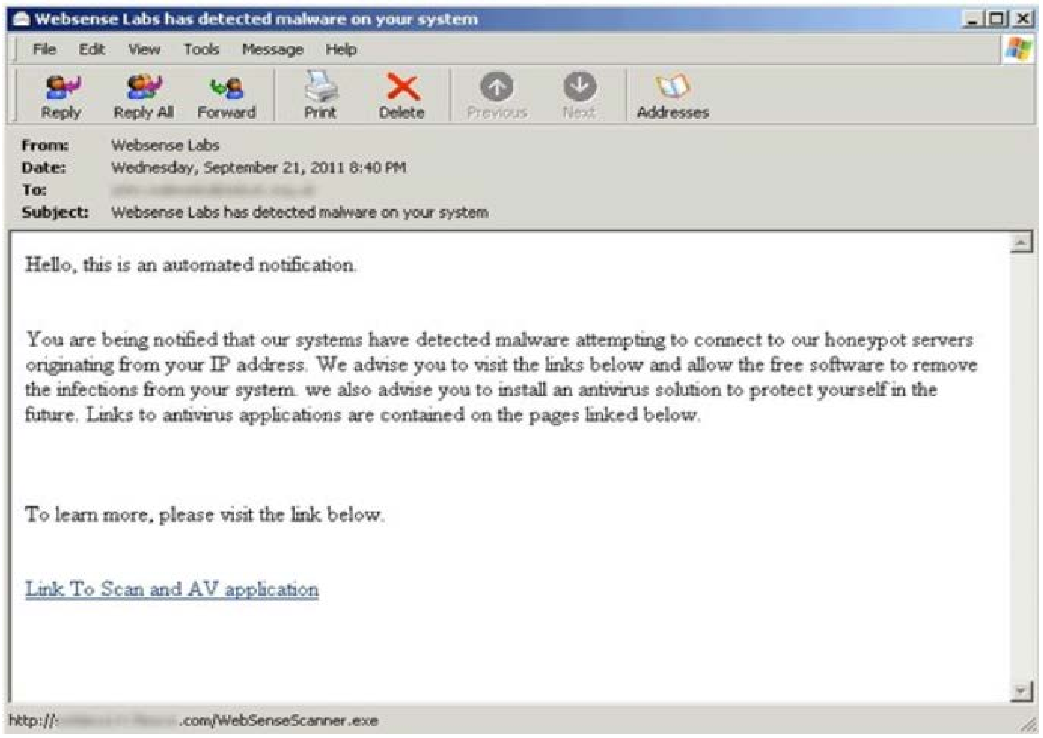
C. ORGANIZATION OF THESIS

Chapter II addresses malicious emails that affect the DOD and the importance of detecting them. Chapter III provides a background on formal specification, natural language, collection of data, rules4business and using the StateRover toolset. Chapter IV explains how to use Hidden Markov Model in runtime monitoring to examine behavioral and temporal patterns over time from collected data to identify hidden data. Chapter V provides the results of validating, generating the HMM, and performing runtime monitoring. Chapter VI identifies shortcomings and recommendations of this thesis and a conclusion.

II. MALICIOUS EMAILS

A. DETECTING MALICIOUS EMAILS BY COLLECTING DATA THROUGH BULK EMAIL OR PHISHING

Many email systems as well as commercial marketing packages allow bulk email that facilitates broad distribution of a message or documents to wide audiences quickly and at low cost. For example, a company can distribute a policy statement to all of its employees or issue a press release to hundreds of media outlets [4]. Since bulk emails are common, most users are unaware of fraudulent actors' intent. The software and mechanisms to produce bulk mail are an easy and inexpensive way to obtain information, often private or sensitive (phishing), damage, disable, or modify the recipients' computer [malware] and/or replicate creating a widening web of disruption (viruses) [5] (see Figure 1).



From: Websense Labs
Date: Wednesday, September 21, 2011 8:40 PM
To:
Subject: Websense Labs has detected malware on your system

Hello, this is an automated notification.

You are being notified that our systems have detected malware attempting to connect to our honeypot servers originating from your IP address. We advise you to visit the links below and allow the free software to remove the infections from your system. we also advise you to install an antivirus solution to protect yourself in the future. Links to antivirus applications are contained on the pages linked below.

To learn more, please visit the link below.

Link To Scan and AV application

http://: .com/WebSenseScanner.exe

Figure 1. Fraudulent Email Example. Source: [5].

Collecting data from bulk email or phishing can help us to categorize the data. With the use of formal validation and verification techniques, we can further capture and target malicious email patterns. As a result, we can see who is targeted by the malicious email.

Undesirable email, for example, spam is sent in mass to an extensive number of individuals on the Internet and is often unwanted, irrelevant, or inappropriate, but it is generally benign. Malicious emails can be similarly distributed, but have nefarious intent. They either prompt recipients to reveal information (account numbers, Social Security numbers, etc.), quietly steal information (e.g., contact lists), or impact computer operation. Often, the fraudulent emails take the guise of a government agency or bank and appear as if they are an official communication. They act, in fact, as a Trojan horse, not being what they appear.

Receiving several of these emails within a week, we will likely perceive these emails to be suspicious. By categorizing them, an organization can more easily decide whether to accept or reject the email coming into their network environment. This especially true when some properties of incoming emails are not deterministically available in the email text; rather, they are probabilistically learned or hidden properties. In this case, reasoning about patterns of emails with hidden artifacts has the potential of providing improved or probabilistic classification. Using the Runtime Monitoring and Verification System, we can provide a way to track activity and meet the requirements to keep our systems safe from malicious emails.

B. DOD TARGETED MALICIOUS EMAILS

Malicious emails not only target individual Internet Service Provider users, but also financial institution and governmental agencies, for example, the Department of Defense (DOD). More refined attacks deploy emails that appear to be indistinguishable to official documents from trusted sources and are therefore a threat to the security of Government officials and government networks [6]. At its worst, national security is put at risk when agencies such as the DOD are targeted.

Spear phishing, in particular, is a significant and widespread type of attack the DOD is battling. In 2006, the JTF-GNO released an article saying that its members have “observed tens of thousands of malicious emails targeting soldiers, sailors, airmen and Marines; U.S. government civilian workers; and DOD contractors, with the potential compromise of a significant number of computers across the DOD” [7]. Therefore, fraudulent actors are targeting government employees to gain more than just account or personal information; they are focused on collecting intelligence which may put our nation in peril. From the accounts that have been compromised, more infiltration of the DOD networks and classified information may occur. However, the true scope is unknown, and some government experts believe that some terrorists, subversives, and foreign countries have already gained a broad range of intelligences on their government targets and seeking additional information to target exactly what they need to next. DOD users are required to digitally sign their emails, but the DOD has not been able to protect personal emails. This thesis seeks to define a means of identifying email threats in a naval and DOD environment.

THIS PAGE INTENTIONALLY LEFT BLANK

III. BACKGROUND

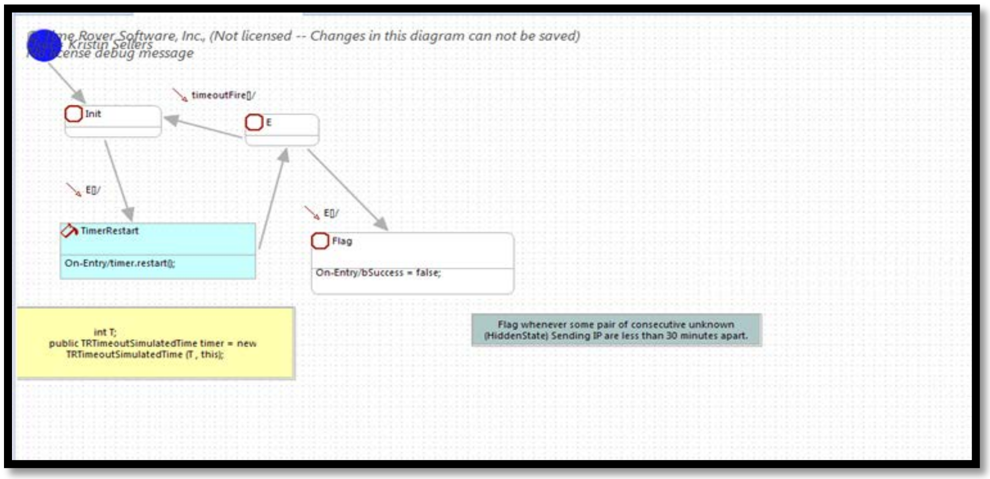
A. NATURAL LANGUAGE FORMAL SPECIFICATIONS AND CORRESPONDING ASSERTION FORMAL SPECIFICATIONS

Consider the following generic natural language (NL) patterns, which are generic rule 9 and rule 11 of the rules4business website:

> Rule 9: *Flag whenever some pair of consecutive E events is less than time T apart.*
>
> Rule 11: *Flag whenever event P with eventual event Q within time after P.*

Figure 2 depicts a statechart-assertion formal specification for rule 9 as designed using the StateRover tool.

A statechart-assertion is standard of UML and is designed to be reusable independent of a specific statechart. The statechart-assertion model, as described by Drusinsky in [2], includes machine representation, with corresponding flowcharting capabilities, defined hierarchy, a Java action language, and a Boolean flag (named bFlag, as show in Figure 2) that indicates if a particular pattern has been flagged. This flag's initial setting is false and becomes a valid value when an assertion is detected. Drusinsky further explains that the statechart-pattern (shown in Figure 2) combines the flowchart and state-machine elements; the statechart flows through the boxes while executing their actions and conditions.


me Rover Software, Inc., (Not licensed -- Changes in this diagram can not be saved)
Kristin Sellers
ense debug message
timeoutFire()/
Init
E
E()/
E()/
TimerRestart
On-Entry/timer.restart();
Flag
On-Entry/bSuccess = false;

```
int T;
public TRTimeoutSimulatedTime timer = new
TRTimeoutSimulatedTime (T, this);
```

Flag whenever some pair of consecutive unknown (HiddenState) Sending IP are less than 30 minutes apart.

Figure 2. A Statechart-Assertion for Requirement Rule 9. Adapted from [8].

As shown in Figure 2, the statechart flows through the Initial flowchart box, executes its actions, and then checks whether the SendingIP transaction is unknown or not. Therefore, if rule 9 has been violated, the statechart-assertion sets the bSuccess flag to false, indicating that the assertion has failed (the Error state) [9].

Since rule 9 and 11 are generic it cannot be used verbatim. NL1 is an instance of generic rule 9. NL2 is an instance of generic rule 11.

NL1. *Flag whenever some pair of consecutive emails whose SendingIP is unknown is less than 30 minutes apart.*

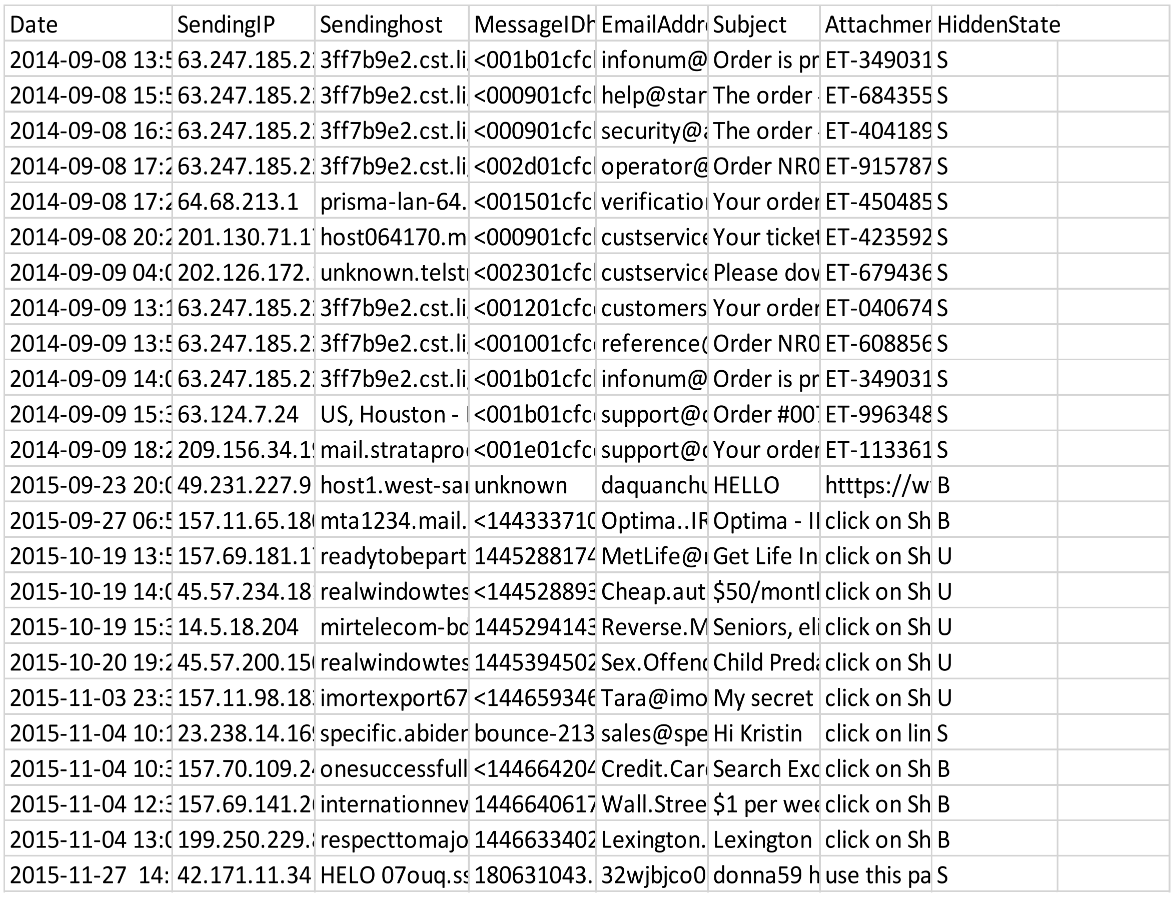
NL2. *Flag when there is a suspicious email within one hour of an email whose Sendinghost is 3ff7b9e2.cst.lightpath.net.*

B. RUNTIME MONITORING

Runtime monitoring (RM) is a technique that allows the user to observe the behavior of the system while it is running. Also, it analyzes the system's current behavior to determine if it satisfies or violates formal specifications. In [10], Drusinsky presents RM tools like TemporalRover and DBRover [10], along with the RM tools, from Havelund and Rosu's paper [11], PaX is an RV tool used to verify Java programs, and RT-Mac [12] chose to use Propositional Linear-time Temporal as their specification language and all of the its extensions, and StateRover [13], whose specification language is deterministic/nondeterministic statechart assertions.

An important aspect of RM is rule validation, where the rule is certified to meet the cognitive expectations of the rule developer. Given that the human cognitive process is often ambiguous and error prone, it is important to test that the formal specification captures the expected behavior to the letter. This is done by manual testing of the formal specification rule. In this thesis, we will use rules4business (described below) to develop formal specification rules; rule validation in rules4business is done by uploading a csv data file called a validation csv file), and checking that the rule indeed flagged when expected to flag, and did not flag when expected not to. Table 1 shows such a validation csv file.

Table 1. Validation CSV File



Date	SendingIP	Sendinghost	MessageIDh	EmailAddr	Subject	Attachmer	HiddenState
2014-09-08 13:5	63.247.185.2	3ff7b9e2.cst.li	<001b01cfcl	infonum@	Order is pr	ET-349031	S
2014-09-08 15:5	63.247.185.2	3ff7b9e2.cst.li	<000901cfcl	help@star	The order	ET-684355	S
2014-09-08 16:3	63.247.185.2	3ff7b9e2.cst.li	<000901cfcl	security@a	The order	ET-404189	S
2014-09-08 17:2	63.247.185.2	3ff7b9e2.cst.li	<002d01cfcl	operator@	Order NR0	ET-915787	S
2014-09-08 17:2	64.68.213.1	prisma-lan-64.	<001501cfcl	verificatio	Your order	ET-450485	S
2014-09-08 20:2	201.130.71.1	host064170.m	<000901cfcl	custservice	Your ticket	ET-423592	S
2014-09-09 04:0	202.126.172.	unknown.telst	<002301cfcl	custservice	Please dov	ET-679436	S
2014-09-09 13:1	63.247.185.2	3ff7b9e2.cst.li	<001201cfc	customers	Your order	ET-040674	S
2014-09-09 13:5	63.247.185.2	3ff7b9e2.cst.li	<001001cfc	reference(	Order NR0	ET-608856	S
2014-09-09 14:0	63.247.185.2	3ff7b9e2.cst.li	<001b01cfcl	infonum@	Order is pr	ET-349031	S
2014-09-09 15:3	63.124.7.24	US, Houston -	<001b01cfc	support@	Order #00	ET-996348	S
2014-09-09 18:2	209.156.34.1	mail.stratapro	<001e01cfc	support@	Your order	ET-113361	S
2015-09-23 20:0	49.231.227.9	host1.west-sa	unknown	daquanchu	HELLO	htttps://w	B
2015-09-27 06:5	157.11.65.18	mta1234.mail.	<14433371	Optima..IR	Optima - I	click on Sh	B
2015-10-19 13:5	157.69.181.1	readytobepart	1445288174	MetLife@	Get Life In	click on Sh	U
2015-10-19 14:0	45.57.234.18	realwindowtes	<144528893	Cheap.aut	$50/mont	click on Sh	U
2015-10-19 15:3	14.5.18.204	mirtelecom-bo	1445294143	Reverse.M	Seniors, eli	click on Sh	U
2015-10-20 19:2	45.57.200.15	realwindowtes	1445394502	Sex.Offenc	Child Preda	click on Sh	U
2015-11-03 23:3	157.11.98.18	imortexport67	<144659346	Tara@imo	My secret	click on Sh	U
2015-11-04 10:1	23.238.14.16	specific.abider	bounce-213	sales@spe	Hi Kristin	click on lin	S
2015-11-04 10:3	157.70.109.2	onesuccessfull	<144664204	Credit.Car	Search Exc	click on Sh	B
2015-11-04 12:3	157.69.141.2	internationnev	1446640617	Wall.Stree	$1 per wee	click on Sh	B
2015-11-04 13:0	199.250.229.	respecttomajo	1446633402	Lexington.	Lexington	click on Sh	B
2015-11-27 14:	42.171.11.34	HELO 07ouq.ss	180631043.	32wjbjco0	donna59 h	use this pa	S

C. FORMAL SPECIFICATION TRADEOFF CUBOID

Traditionally, formal specifications are used for Validation and Verification (V&V). Verification means to ensure a product is built correctly. As Meseguer and Preese states [14], “Validation is a process aimed at demonstrating that a system meets the user’s true requirements--often called ‘building the right system’” [14]. To select a validation and verification technique that is appropriate for detecting temporal patterns of malicious emails, we used the visual tradeoff space from Drusinsky, Michael, and Shing’s paper in [15], which compares three predominant formal validation and verification techniques. Noted in Drusinsky, Michael, and Shing’s paper, the three techniques include theorem proving, model checking, and runtime monitoring.

The “cube” is a three dimensional comparison known as the formal validation and verification tradeoff cube; it is illustrated in Figures 3 and 4 [15]. The tradeoff cubes depict the coverage and cost of each of the three techniques. The three dimensions of the coverage and cost cubes are (i) specification dimension—the technique’s capacity to specify complex properties, (ii) the efficiency of verification dimension, and (iii) the complexity of programs that can be verified.

Ultimately, we chose RM as the best method of monitoring malicious emails because we are not concerned with the verification and program dimensions of the cube. When monitoring for patterns of malicious emails, there is no underlying program to verify.

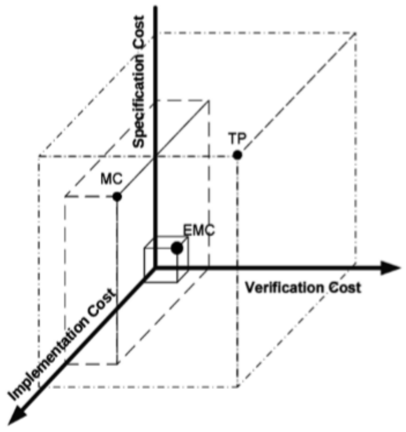


Figure 3. Cost Space. Source: [15].

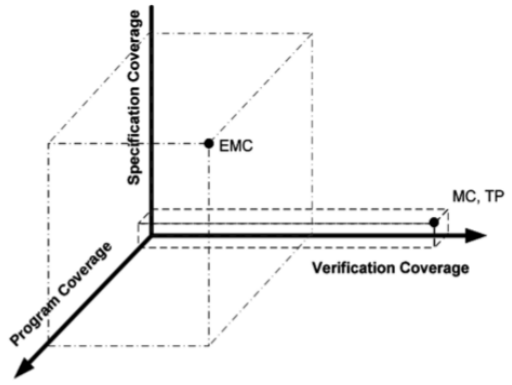


Figure 4. Coverage Space. Source: [15].

D. TRANSLATING NATURAL LANGUAGE TO FORMAL SPECIFICATION

In software engineering, formal specifications are scientifically based procedures that help with the implementation of systems and software. They are used to portray a system, to examine its conduct, and to help in its configuration by confirming key properties of interest. These specifications are formal in the sense that they help improve the clarity and precision of requirements. So, the question is asked, "why convert natural language to formal specifications?"

Natural language (NL) is inherently ambiguous, rendering accurate specification problematic [16]. However, formal specifications allow us to convey the exact intent of the natural language requirement. Essentially, a formal specification is meant to pinpoint particular information that the user seeks to extract from the natural language. Drusinsky, Michael, and Shing's paper [17] presents patterns for ensuring that formal specifications catch the intent of underlying natural language requirements [18].

For example, we give a generalization of how natural language can be ambiguous. *No restaurants will allow smoking inside.* Here *no* can qualify the rest of the sentence, meaning thereby there is not a restaurant that will allow smoking inside. On the other hand, it can qualify only the phrase *restaurant*, meaning thereby there *are restaurants* designated as *no restaurants*, which, however, allow *smoking inside*. By using formal specification, it makes sure it is doing exactly what it means to do.

E. HIDDEN MARKOV MODELS

Markov Models are stochastic models that are used in randomly alerting systems. As described in Rabiner [19], HMM components are: (i) a set of states, (ii) observations made in those states, (iii) state transition probabilities, and (iv) initial state distribution. HMM is a statistical model where the set of states are not fully visible, while its state outputs are visible. Figure 5 illustrates an example of an HMM. Its state set X, set of observables y, state transitions matrix A, and the matrix B of emission probabilities are all depicted visually.

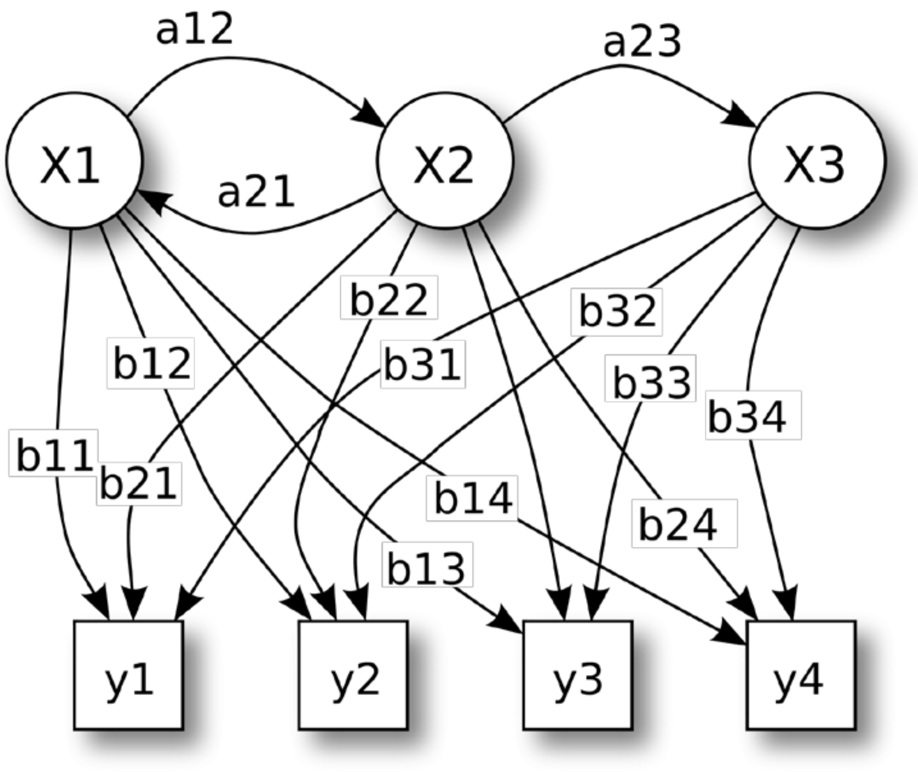


Figure 5. Hidden Markov Model. Source [20].

F. THE HMM ALPHA METHOD

The alpha method (also known as the Forward algorithm) is a well-known technique for calculating the probability an HMM reaches each one of its state at time *i*, ($0 \leq i \leq T$), given an observation sequence of length T.

Specially, $\alpha_{t+1}(j) = [\sum_{i=1...N} \alpha_t(i) a_{ij}] b_j O_{t+1}), 1 \leq t \leq T-1, 1 \leq j \leq N$, with the initialization: $\alpha_1(j) = \pi_j b_j(O_1)$. Note that $P(O_1 O_2 ... O_t|\lambda) = \sum_{i=1...N} \alpha_t(i)$. α' is the normalized version of α: $\alpha'_t(j) = P(q_t = s_i|O_1 O_2 ... O_t \lambda,)$.

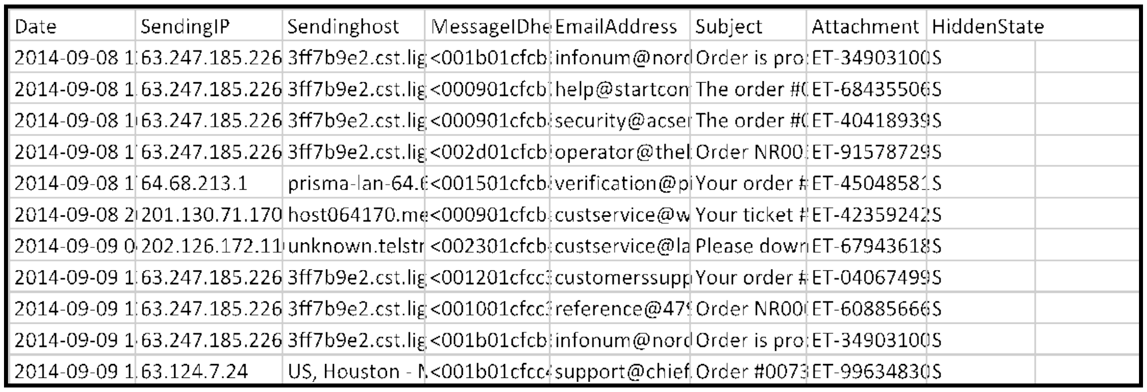
G. COLLECTION OF DATA

We used data from Naval Postgraduate School Information Technology and Communications Services (ITACS) and bulk and phishing emails from a personal account. We read through the emails to gather specific information such as date and time, the sending IP and host address, whether the email had an attachment or link, and so on. This information was stored in a CSV file; to do so, we acted as the expert pulling information from the emails. In fact, we created three csv file versions (each being a table): a validation table, learning phase, and runtime table. Tables 1 through 3 show snippets of these csv file, respectively. In Table 4, we show the meaning of table columns.

Note that the three csv file versions do not have the same schema:

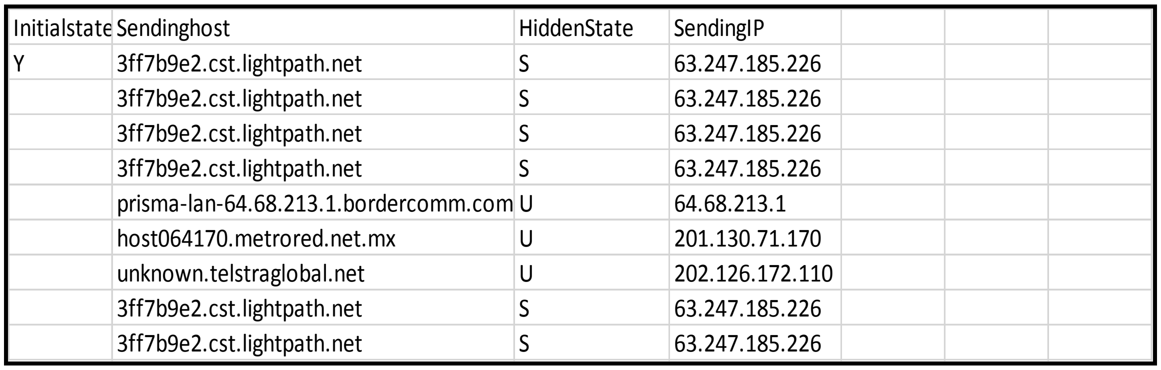
1. The validation csv file consists of visible data we were able to gather from the email and its hidden state column is populated (i.e., it is visible—not hidden).
2. The learning phase csv file is used to learn the HMM; it also contains a hidden-state column, populated by a subject-matter expert.
3. The runtime csv file includes all the real data except the hidden state column—the HMM is used in run-time instead of that column. In Chapter V, we will see the results of using these three csv files.

Table 2. Snippet of Validation CSV File



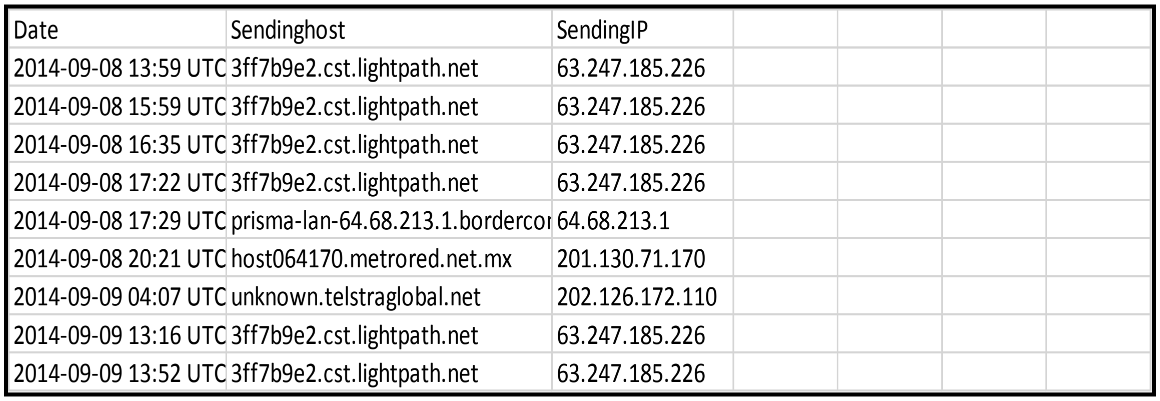
Date	SendingIP	Sendinghost	MessageIDhe	EmailAddress	Subject	Attachment	HiddenState
2014-09-08 1	63.247.185.226	3ff7b9e2.cst.lig	<001b01cfcb	infonum@nord	Order is pro	ET-34903100	S
2014-09-08 1	63.247.185.226	3ff7b9e2.cst.lig	<000901cfcb	help@startcon	The order #	ET-68435506	S
2014-09-08 1	63.247.185.226	3ff7b9e2.cst.lig	<000901cfcb	security@acser	The order #	ET-40418939	S
2014-09-08 1	63.247.185.226	3ff7b9e2.cst.lig	<002d01cfcb	operator@thel	Order NR00	ET-91578729	S
2014-09-08 1	64.68.213.1	prisma-lan-64.6	<001501cfcb	verification@pi	Your order #	ET-45048581	S
2014-09-08 2	201.130.71.170	host064170.me	<000901cfcb	custservice@w	Your ticket #	ET-42359242	S
2014-09-09 0	202.126.172.11	unknown.telstr	<002301cfcb	custservice@la	Please down	ET-67943618	S
2014-09-09 1	63.247.185.226	3ff7b9e2.cst.lig	<001201cfcc	customerssupp	Your order #	ET-04067499	S
2014-09-09 1	63.247.185.226	3ff7b9e2.cst.lig	<001001cfcc	reference@47	Order NR00	ET-60885666	S
2014-09-09 1	63.247.185.226	3ff7b9e2.cst.lig	<001b01cfcb	infonum@nord	Order is pro	ET-34903100	S
2014-09-09 1	63.124.7.24	US, Houston - N	<001b01cfcc	support@chief.	Order #0073	ET-99634830	S

Table 3. Snippet of Learning Phase CSV File



Initialstate	Sendinghost	HiddenState	SendingIP
Y	3ff7b9e2.cst.lightpath.net	S	63.247.185.226
	3ff7b9e2.cst.lightpath.net	S	63.247.185.226
	3ff7b9e2.cst.lightpath.net	S	63.247.185.226
	3ff7b9e2.cst.lightpath.net	S	63.247.185.226
	prisma-lan-64.68.213.1.bordercomm.com	U	64.68.213.1
	host064170.metrored.net.mx	U	201.130.71.170
	unknown.telstraglobal.net	U	202.126.172.110
	3ff7b9e2.cst.lightpath.net	S	63.247.185.226
	3ff7b9e2.cst.lightpath.net	S	63.247.185.226

Table 4. Snippet of Runtime CSV File



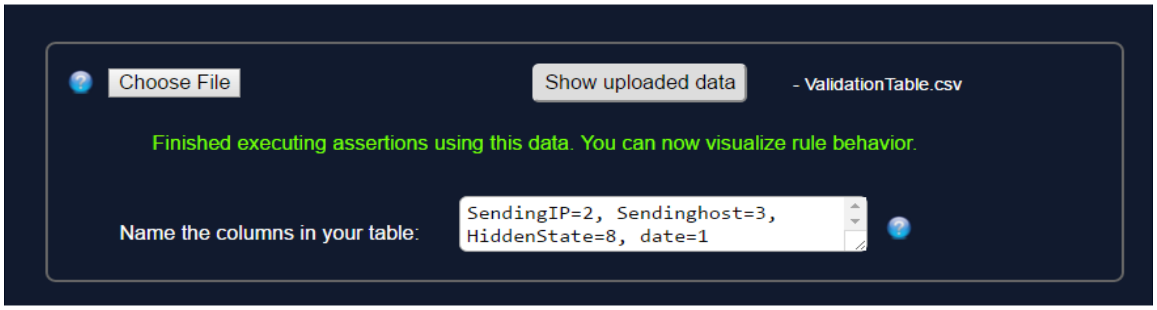
Date	Sendinghost	SendingIP
2014-09-08 13:59 UTC	3ff7b9e2.cst.lightpath.net	63.247.185.226
2014-09-08 15:59 UTC	3ff7b9e2.cst.lightpath.net	63.247.185.226
2014-09-08 16:35 UTC	3ff7b9e2.cst.lightpath.net	63.247.185.226
2014-09-08 17:22 UTC	3ff7b9e2.cst.lightpath.net	63.247.185.226
2014-09-08 17:29 UTC	prisma-lan-64.68.213.1.bordercor	64.68.213.1
2014-09-08 20:21 UTC	host064170.metrored.net.mx	201.130.71.170
2014-09-09 04:07 UTC	unknown.telstraglobal.net	202.126.172.110
2014-09-09 13:16 UTC	3ff7b9e2.cst.lightpath.net	63.247.185.226
2014-09-09 13:52 UTC	3ff7b9e2.cst.lightpath.net	63.247.185.226

Table 5. Meaning of Columns

Columns	Meaning
Date/time	The date and time when the email was received
SendingIP	The sender’s IP address, where the email is coming from
Sendinghost	Remote domain that send emails to your server
MessageID	Unique ID for Internet messages
EmailAddress	Who the email is coming from
Subject	A title that alerts to read or delete
Attachment	A file, link, malware, etc., that sent along with the email

H. RULES4BUSINESS

Rules4Business (R4B) is a website that allows users to create rules based on events and timing patterns. The rules are a way of analyzing and verifying the behavior of the patterns in the csv file. The user can use R4B to choose, customize statechart assertions, and edit instances of the generic rule. In R4B, users have two interfaces for customizing and validating assertions. First, users select a rule according the NL specifications. On the second page of R4B, users upload the validation csv file, explained in Section G, with the required columns to be able to validate assertions. Figure 6 shows an example of how to specify the column indexes before uploading the csv file. We specify the column indexes from the columns in our csv file that we want R4B to validate.



Finished executing assertions using this data. You can now visualize rule behavior.

Name the columns in your table:

```
SendingIP=2, Sendinghost=3,
HiddenState=8, date=1
```

Figure 6. Name of the Columns in R4B. Source: [8].

In this thesis, we chose to customize two generic R4B rules: rule 9 and rule 11; their instances are shown in Table 6. Figures 7 and 8 show the corresponding UML-statecharts for each rule. Customization is done by specifying specific attributes for the generic attributes within the generic rules. The resulting instances are shown in Table 6.

Table 6. Instances of Rule 9 and Rule 11. Adapted from [8].

Rule 9	Generic Pattern	Flag whenever some pair of consecutive E events is less than time T apart
	Custom properties (Events and Limits)	E=HiddenState===“U,” Time bounds: T=30, Time units: minutes
	Description	Flag whenever some pair of consecutive unknown SendingIP are less than 30 minutes apart.
Rule 11	Generic Pattern	Flag whenever event P with eventual event Q within time T after P.
	Custom properties (Events and Limits)	P= Sendinghost.indexOf(“3ff7b9e2.cst.lightpath.net”)>=0, Q=HiddenState===“S,” Time bounds: T=1, Time units: hours
	Description	Flag when there is a suspicious email within one hour of an email from 3ff7b9e2.cst.lightpath.net

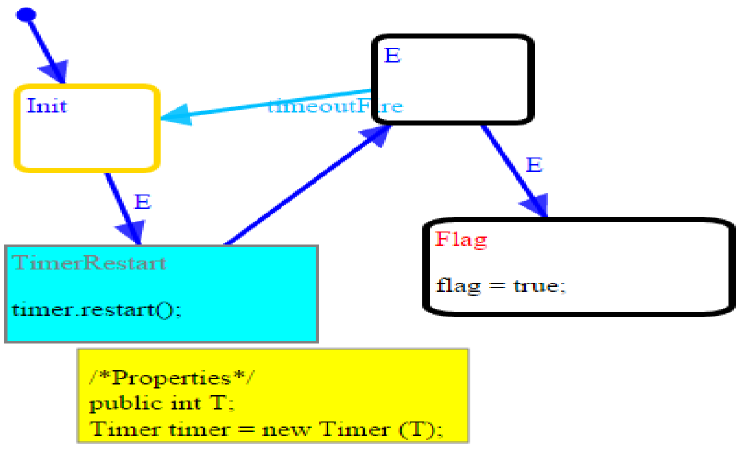


Figure 7. Rules4business Rule 9 UML-Statechart. Source: [8].

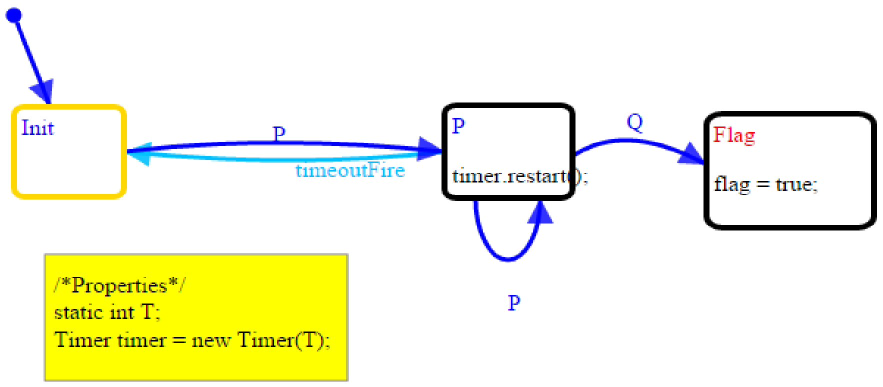


Figure 8. Rules4business Rule 11 UML-Statechart. Source: [8].

In this thesis, Rules4business is used for the specification and validation of natural language and formal specification rules. It checks an uploaded data (csv format) file against the rule instance (i.e., the pattern), thereby performing runtime monitoring (RM) using the formal specification statechart assertion. The output from this operation shows, visually, where the rule has flagged or not flagged. Results will be shown in Chapter V. Note that the rules we developed using rules4business are used in the results chapter, by applying them to data that is partially hidden.

I. THE STATEROVER TOOLSET

In this research, the StateRover is used as part of the code generation process. The code generation process is implemented by the *dtracg* tool (see Chapter V.B), which relies on code generated from the StateRover. There is no other reason for using the StateRover in this research, other than this purely technical reason; therefore, uninterested readers can jump to Chapter V.B.

According to Drusinsky [9], the StateRover used in this research "extends the statechart diagrammatic notation with Java as an action language, resulting in a Turing-equivalent notation." Before using the StateRover code generator, we perform validation testing to ensure that assertion drawn in the StateRover is the behaviorally equivalent to the rule taken from rules4business.

IV. OVERVIEW OF HYBRID RM: RM WITH HIDDEN DATA

A. ARCHITECTURE

In Chapter III, we overviewed deterministic RM. In this chapter we introduce a recent architecture that enables RM of data streams that contain hidden artifacts, using HMM's in the loop [21]. This architecture is depicted in Figure 9.

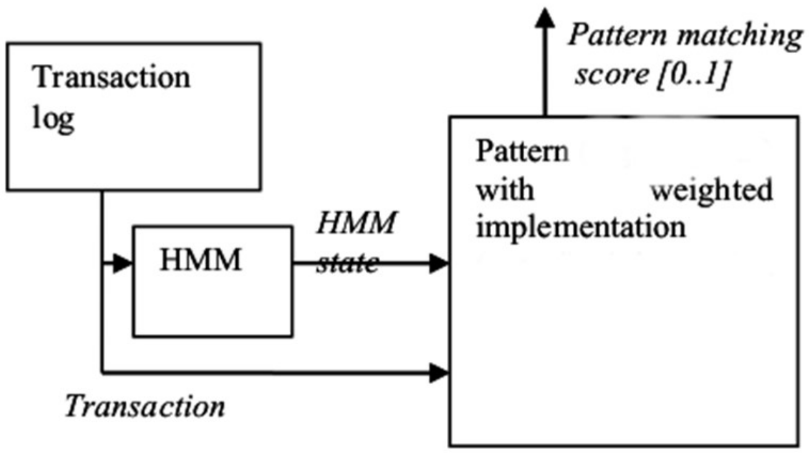

Transaction log
HMM
HMM state
Pattern with weighted implementation
Pattern matching score [0..1]
Transaction

Figure 9. Pattern Matching Architecture for Malicious Emails. Source: [21].

The manner in which the HMM is used as part of the RM system is as follows. In runtime, transaction data (being email data, in our case, as described in section V) is fed into the HMM, which executes an iterative probability estimation algorithm [21]. Using the Alpha-method described in Chapter III.E, the HMM outputs the stream of pairs <HMM-state, associated state visitation probability>. This stream is used as an input to the rule's implementation code, code that implements a special weighted RM algorithm, described below.

B. ALGORITHM FOR RM WITH HIDDEN DATA

RM monitor of Figure 9 performs RM of a data stream that contains both visible and hidden data. The outline of the algorithm is as follows [2]. The monitor's input is a sequence of pairs: {K_1,P_1},{K_2,P_2}, {K_3,P_3}…{K_N,P_N}. K_i is an event that is visible (e.g., *Sendinghost* and *SendingIP*) or hidden (e.g., HiddenState column). P_i is the probability K_i. In general, K_i is given in UML format: event$_i$ [condition$_i$], either could be visible or hidden.

The runtime behavior of the monitor is as follows. Each assertion contains a collection of one or more instances called configurations. Collection is labeled as Col and the configuration as Conf. Each Conf has a present state PS(Conf) and probability value called P(Conf) a probability measure indicating the weight of that Conf within Col. Upon startup Col contains a single Conf whose probability is 1. In cycle *i*, if P_i=1, the Conf acts like a traditional state machine, causing PS(Conf) to change. If $P_i \neq 1$, i.e., event$_i$ is hidden, then the Conf is substituted by two configurations called Conf1 and Conf2. Probabilities and states of Conf1 and Conf2 as follows:

- If event$_i$ is hidden,

  P(Conf1)=P(Conf)*P_i and P(Conf2)=1-P(Conf1)

  PS(Conf1) is the next state decided by transition, if event fired. If not then, PS(Conf) assigned to PS(Conf2).

- If condition$_i$ is hidden,

  P(condition$_i$) is calculated according to the constitutive components. For instance, if condition$_i$ is HiddenState=M || HiddenState=S, P(condition$_i$)=P(HiddenState=M) + P(HiddenState=S). And then P(Conf1)=P(Conf)*P(condition$_i$) and P(Conf2)=1-P(Conf1)

  PS(Conf1) and PS(Conf2) are calculated as condition$_i$ is true and false.

Configurations that have same present state are joined in a one configuration as Conf$_{combined}$ by summing all P'(Conf).

The statechart assertions proclaims the probability of violation of its corresponding requirements also known as probability of failure (POF) [2] by computing the weight of all Conf's that are in the Error state (also known as the Flag state).

C. WORKFLOW

In this thesis, we show that when monitoring for patterns of malicious emails, there is no underlying program to verify the system correctness. We are using a powerful formal specification that allow RM to detect these malicious emails. We are going to combine HMM consisting of hidden data and RM of statechart assertions. HMM is used for deducting categorized hidden data such as S, U, or B emails by using observable data and sequences. Figure 10 depicts a workflow chart using RM with hidden data.

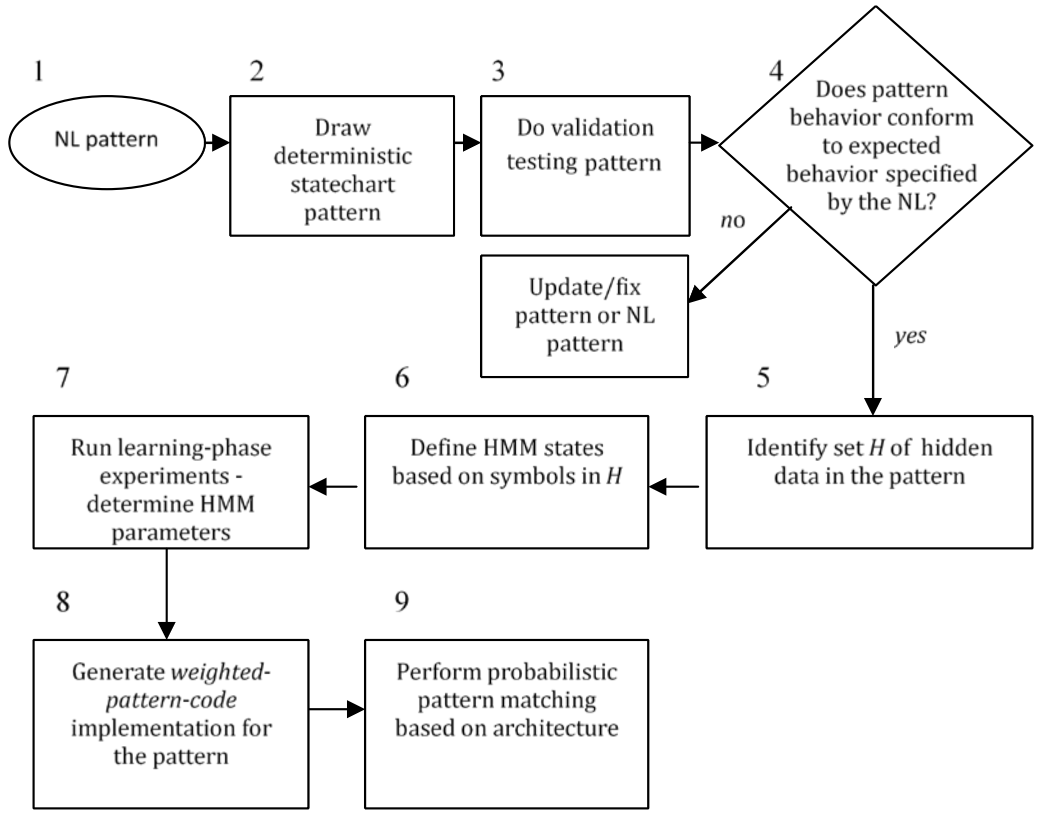

1
NL pattern
2
Draw deterministic statechart pattern
3
Do validation testing pattern
4
Does pattern behavior conform to expected behavior specified by the NL?
no
Update/fix pattern or NL pattern
yes
5
Identify set *H* of hidden data in the pattern
6
Define HMM states based on symbols in *H*
7
Run learning-phase experiments - determine HMM parameters
8
Generate *weighted-pattern-code* implementation for the pattern
9
Perform probabilistic pattern matching based on architecture

Figure 10. Workflow for Developing Pattern Matching with Hidden Information. Source: [21].

D. HMM LEARNING

In the learning phase, an HMM is created from learning data, being a learning phase csv file in our case.

In general, the learning phase csv file contains visible columns and one special column, the HiddenState column, that is manually populated by a subject-matter expert (SME). For simplicity, let’s assume there is a single visible column k. Let k, s, and N be the visible output column, hidden state column, and the total number of rows, respectively; let k_i and s_i are the values of the visible output and hidden state columns in row i. In Drusinsky’s paper [2], an HMM is derived from these artifacts as follows:

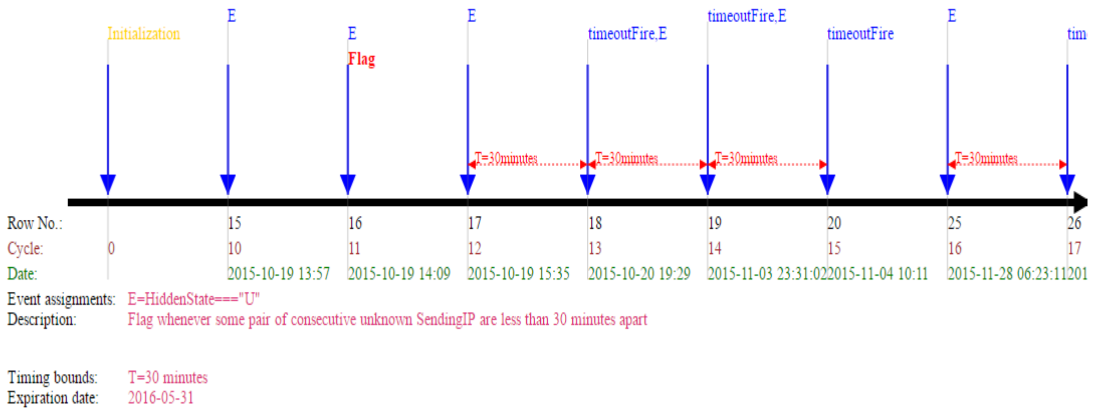
- The HMM state transition probability is calculated by dividing the number of specific transitions to N-1 (total number of transitions in the csv file). For example, suppose there are 15 transitions from the suspicious (S) state to unknown (U) state and N is 31, then the probability of the S->U transition is 15/30=0.5
- For every hidden state S and every observable O, the probability of O being emitted in S is the number of rows i where k_i =O and s_i =S.
- An initial-state probability is assigned to every hidden state S; it is denoted $\pi(S)$. $\pi(S)$ is calculated number of rows of the spreadsheet that contain S and is also marked as an initial state, divided by the number of rows that are marked as an initial state row.

V. RESULTS: PROOF OF CONCEPT

In this chapter, we demonstrate the process of monitoring and validating the sequence and temporal behavior of detecting malicious email. We also demonstrate a hybrid system where RM combined with an HMM is able to monitor both visible and hidden data.

A. DETERMINISTIC RULE DEVELOPMENT

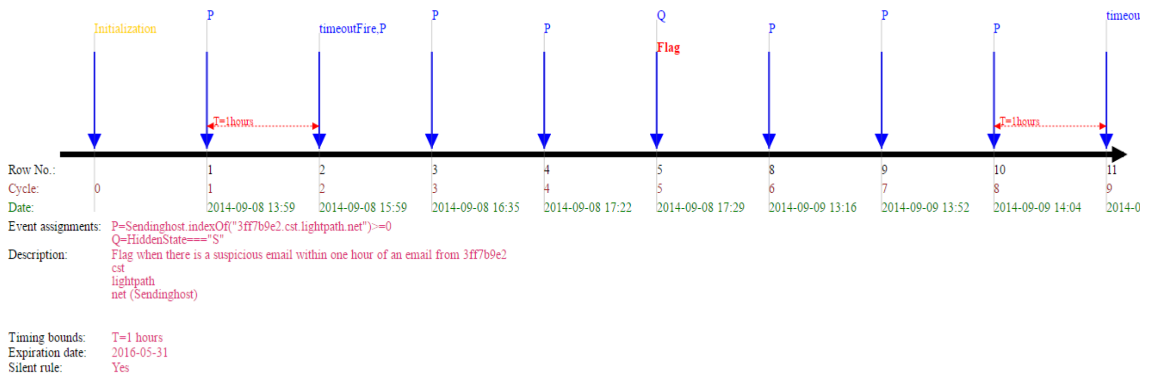
In Chapter III.D, we discussed how to create and validate the rules using R4B website. In our validation phase, we validated rules 9 and 11. Rule 9 determines whether the emails are less than 30 minutes apart if the sending IP is an unknown (U) threat. As discussed in Chapter III.D, rule 11 determines whether a suspicious email within one hour is from a specific Sending host address. In Figures 11 and 12, we show whether the results were what we expected. Figures 13 and 14 show each rule reaching the Flag state.



```
Row No.:                15                16                17                18                19                  20                25                  26
Cycle:         0        10                11                12                13                14                  15                16                  17
Date:                   2015-10-19 13:57  2015-10-19 14:09  2015-10-19 15:35  2015-10-20 19:29  2015-11-03 23:31:022015-11-04 10:11  2015-11-28 06:23:11201
Event assignments:  E=HiddenState==="U"
Description:        Flag whenever some pair of consecutive unknown SendingIP are less than 30 minutes apart

Timing bounds:      T=30 minutes
Expiration date:    2016-05-31
```

Figure 11. Capture of Rule 9 Flag Timeline. Source: [8].

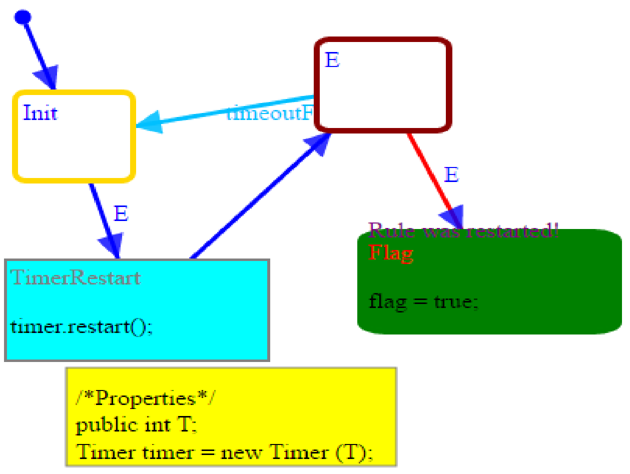


Row No.:		1	2	3	4	5	8	9	10	11
Cycle:	0	1	2	3	4	5	6	7	8	9
Date:		2014-09-08 13:59	2014-09-08 15:59	2014-09-08 16:35	2014-09-08 17:22	2014-09-08 17:29	2014-09-09 13:16	2014-09-09 13:52	2014-09-09 14:04	2014-0

Event assignments:	P=Sendinghost.indexOf("3ff7b9e2.cst.lightpath.net")>=0
	Q=HiddenState==="S"

Description:	Flag when there is a suspicious email within one hour of an email from 3ff7b9e2 cst lightpath net (Sendinghost)
Timing bounds:	T=1 hours
Expiration date:	2016-05-31
Silent rule:	Yes

Figure 12. Capture of Rule 11 Timeline Source: [8].



```
timer.restart();
```

```
flag = true;
```

```
/*Properties*/
public int T;
Timer timer = new Timer (T);
```

Figure 13. Rule 9 Reaching Flag State. Source: [8].

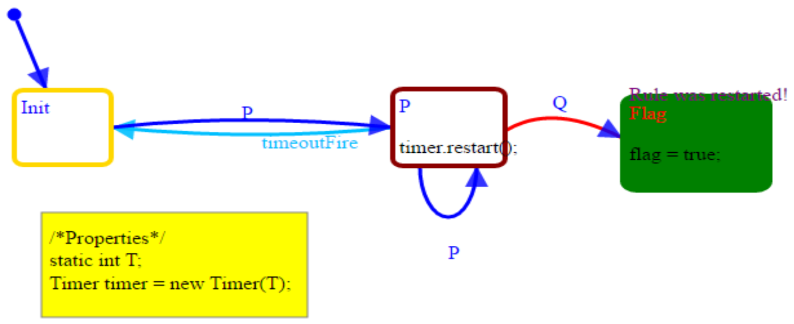


Figure 14. Rule 11 Reaching Flag State. Source: [8].

Rule 9 flagged an unknown email within 30 minutes, and rule 11 flagged a suspicious email within one hour from the specific Sending host. Therefore, we validated both rules and found that both flagged what we expected them to or not.

B. STATEROVER RULE CREATION AND CODE GENERATION

In Chapter III, Section I, we discussed the purpose of the StateRover, which we used in our process to save development time and money when creating the DTRA toolset.

In this section, we show the conversion of R4B diagrams to StateRover diagrams. A snapshot of the statechart assertion of rule 11 is shown in Figure 15. The statechart assertion starts with the initial state, and then the events transition between states. The final state is known as the flag state, which lets us know whether the assertion succeeds or fails. Whenever the StateRover reaches the final state, it yields a *false* value to *bSuccess* because the assertion sees a flagged event.

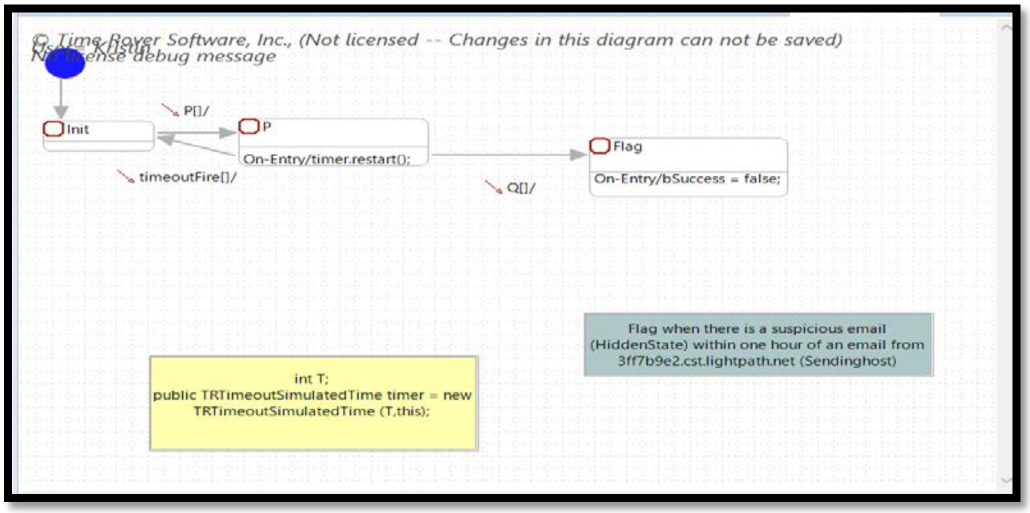
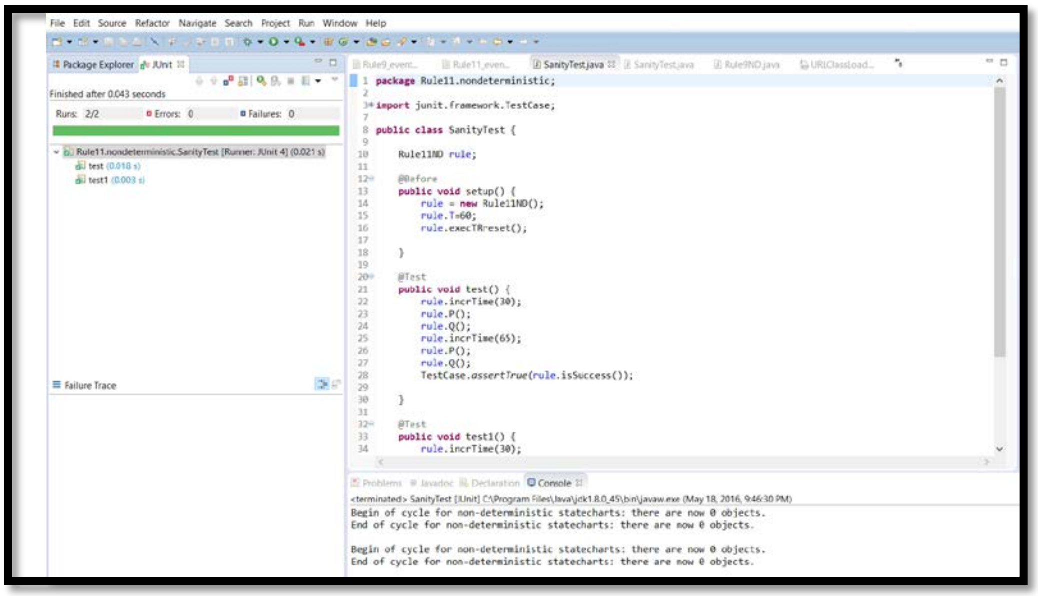

© Time Rover Software, Inc., (Not licensed -- Changes in this diagram can not be saved)
No license debug message
Init
P[]/
P
On-Entry/timer.restart();
timeoutFire[]/
Q[]/
Flag
On-Entry/bSuccess = false;
Flag when there is a suspicious email (HiddenState) within one hour of an email from 3ff7b9e2.cst.lightpath.net (Sendinghost)
int T;
public TRTimeoutSimulatedTime timer = new TRTimeoutSimulatedTime (T,this);

Figure 15. Rule 11 Statechart Assertion. Adapted from [8].

The StateRover implements two steps process to verify the rules. First, StateRover generates Java code based on our statechart diagrams. Second, we run a JUnit test to verify that the StateRover has the same behavior patterns for each statechart assertion as in R4B. Figure 16 shows a successfully run JUnit sanity test.



```
Finished after 0.043 seconds
Runs: 2/2    Errors: 0    Failures: 0
Rule11.nondeterministic.SanityTest [Runner: JUnit 4] (0.021 s)
    test (0.018 s)
    test1 (0.003 s)
```

```java
package Rule11.nondeterministic;

import junit.framework.TestCase;

public class SanityTest {

    Rule11ND rule;

    @Before
    public void setup() {
        rule = new Rule11ND();
        rule.T=60;
        rule.execTRreset();

    }

    @Test
    public void test() {
        rule.incrTime(30);
        rule.P();
        rule.Q();
        rule.incrTime(65);
        rule.P();
        rule.Q();
        TestCase.assertTrue(rule.isSuccess());

    }

    @Test
    public void test1() {
        rule.incrTime(30);
```

```
<terminated> SanityTest [JUnit] C:\Program Files\Java\jdk1.8.0_45\bin\javaw.exe (May 18, 2016, 9:46:30 PM)
Begin of cycle for non-deterministic statecharts: there are now 0 objects.
End of cycle for non-deterministic statecharts: there are now 0 objects.

Begin of cycle for non-deterministic statecharts: there are now 0 objects.
End of cycle for non-deterministic statecharts: there are now 0 objects.
```

Figure 16. JUnit Sanity Test

C. (AUTOMATICALLY) LEARNING THE HMM

The first step in this learning phase is to define the HMM's state set. Using the information from our email data set, we determined that the HMM should contain the following three states, reflecting three types of emails:

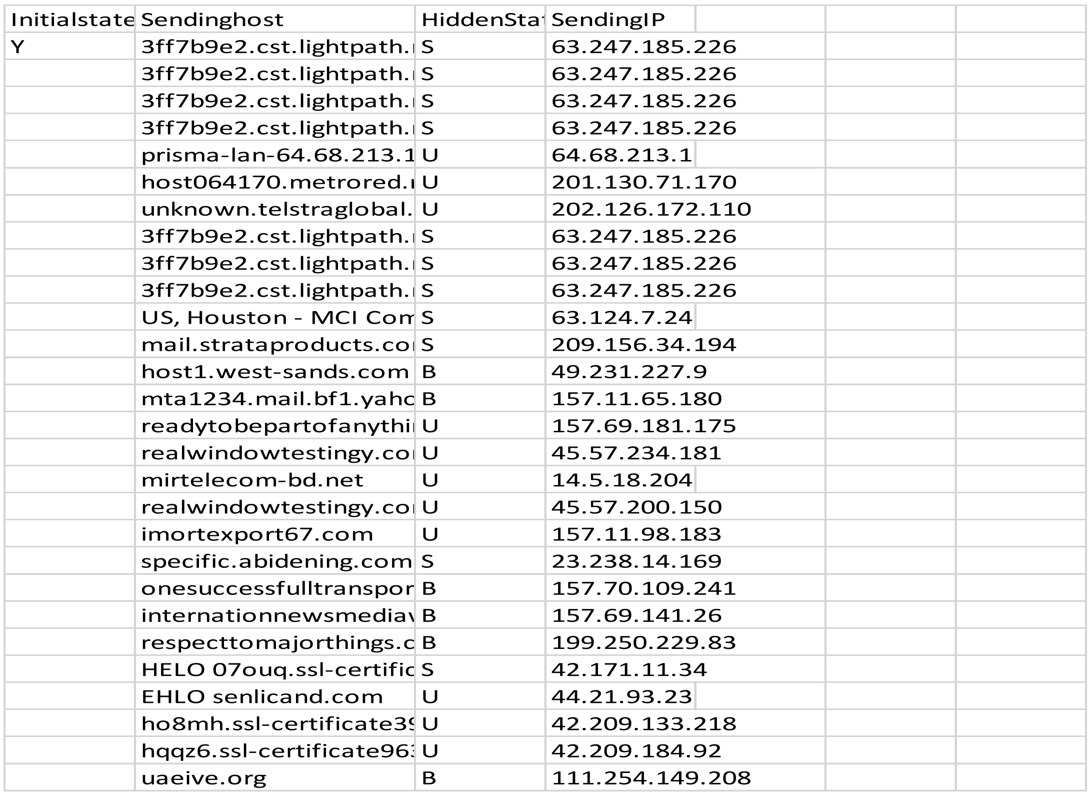
- Suspicious (S)—indicates that the subject-matter expert (SME) witnessed some suspicious behavior associated with some of the other datum in this row, such as a suspicious geographic location of an IP.
- Unknown (U)—indicates that the SME could not determine whether the email is suspicious of not.
- Benign (B).

Some of the email artifacts used to make the abovementioned state determination were: email date and time, source IP address, internal links, and attachments. Figure 19 shows a snippet of our learning phase csv file. Two annotation examples are:

- Row 12 is considered to be suspicious because looking at the email pattern we see that the sender has sent out from the same sending host or based on the geographical location of the sending IP.
- Row 15 is considered to be benign because based off of the geographical location of the sending IP, it was a relatively safe zone.

HMM learning was performed based on the technique described in Chapter IV.D, using the corresponding *dtrahmm* tool [22]. The learning phase uses a special version of csv data file called a *learning table*; Table 8 depicts a snippet of the learning table used in this thesis.

Table 8. Learning CSV File

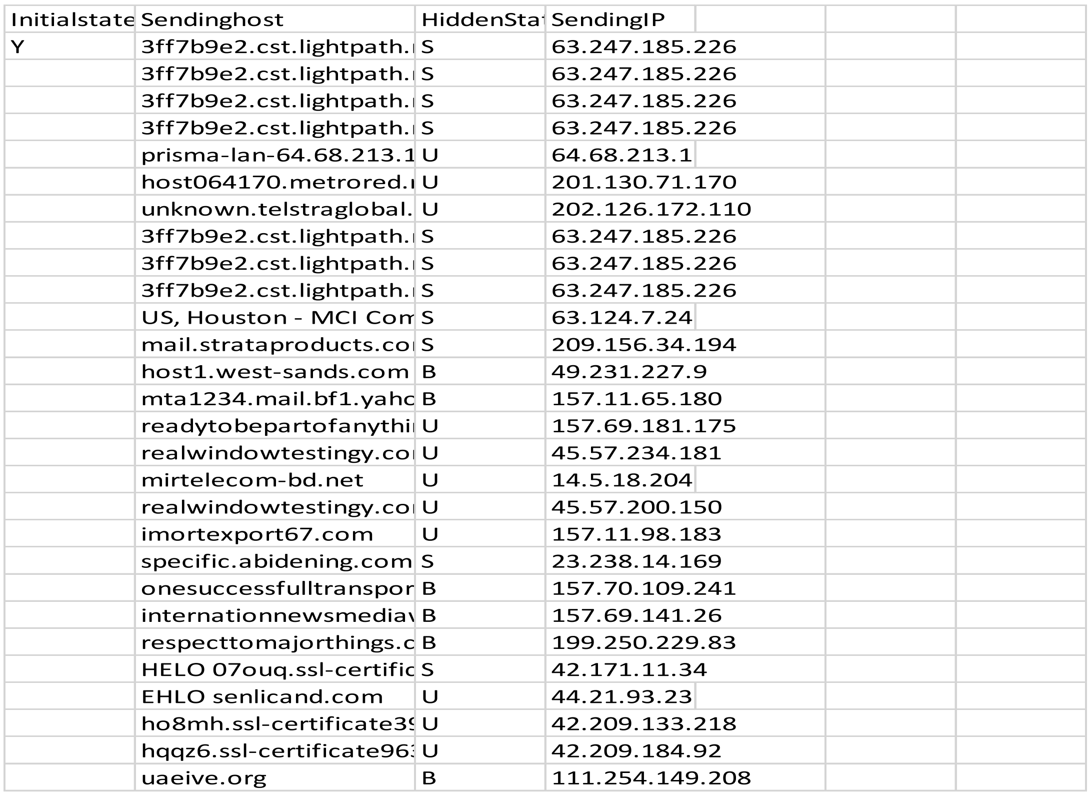


Initialstate	Sendinghost	HiddenSta	SendingIP
Y	3ff7b9e2.cst.lightpath.	S	63.247.185.226
	3ff7b9e2.cst.lightpath.	S	63.247.185.226
	3ff7b9e2.cst.lightpath.	S	63.247.185.226
	3ff7b9e2.cst.lightpath.	S	63.247.185.226
	prisma-lan-64.68.213.1	U	64.68.213.1
	host064170.metrored.	U	201.130.71.170
	unknown.telstraglobal.	U	202.126.172.110
	3ff7b9e2.cst.lightpath.	S	63.247.185.226
	3ff7b9e2.cst.lightpath.	S	63.247.185.226
	3ff7b9e2.cst.lightpath.	S	63.247.185.226
	US, Houston - MCI Com	S	63.124.7.24
	mail.strataproducts.co	S	209.156.34.194
	host1.west-sands.com	B	49.231.227.9
	mta1234.mail.bf1.yahc	B	157.11.65.180
	readytobepartofanythi	U	157.69.181.175
	realwindowtestingy.co	U	45.57.234.181
	mirtelecom-bd.net	U	14.5.18.204
	realwindowtestingy.co	U	45.57.200.150
	imortexport67.com	U	157.11.98.183
	specific.abidening.com	S	23.238.14.169
	onesuccessfulltranspor	B	157.70.109.241
	internationnewsmedia	B	157.69.141.26
	respecttomajorthings.c	B	199.250.229.83
	HELO 07ouq.ssl-certific	S	42.171.11.34
	EHLO senlicand.com	U	44.21.93.23
	ho8mh.ssl-certificate3	U	42.209.133.218
	hqqz6.ssl-certificate96	U	42.209.184.92
	uaeive.org	B	111.254.149.208

The last phase to generate HMM is to run the command for generating hmm.json file which includes the quantized visible data for the hidden states as shown in Table 9. The hmm.json is the output from the HMM parameters. The HMM parameters learned in this phase are:

- Matrix A, the state transition probability matrix, shown in Table 10.
- Matrix B, observable emission probability matrix (the probability of an observable O being emitted in state S), shown in Table 11.
- Π(*i*), Initial state probability; We assume the following initial state probability: <1, 0, 0> for <S, U, B>, respectively.

Table 9. Learning Phase CSV File



Initialstate	Sendinghost	HiddenSta	SendingIP			
Y	3ff7b9e2.cst.lightpath.	S	63.247.185.226			
	3ff7b9e2.cst.lightpath.	S	63.247.185.226			
	3ff7b9e2.cst.lightpath.	S	63.247.185.226			
	3ff7b9e2.cst.lightpath.	S	63.247.185.226			
	prisma-lan-64.68.213.1	U	64.68.213.1			
	host064170.metrored.	U	201.130.71.170			
	unknown.telstraglobal.	U	202.126.172.110			
	3ff7b9e2.cst.lightpath.	S	63.247.185.226			
	3ff7b9e2.cst.lightpath.	S	63.247.185.226			
	3ff7b9e2.cst.lightpath.	S	63.247.185.226			
	US, Houston - MCI Com	S	63.124.7.24			
	mail.strataproducts.co	S	209.156.34.194			
	host1.west-sands.com	B	49.231.227.9			
	mta1234.mail.bf1.yahc	B	157.11.65.180			
	readytobepartofanythi	U	157.69.181.175			
	realwindowtestingy.co	U	45.57.234.181			
	mirtelecom-bd.net	U	14.5.18.204			
	realwindowtestingy.co	U	45.57.200.150			
	imortexport67.com	U	157.11.98.183			
	specific.abidening.com	S	23.238.14.169			
	onesuccessfulltranspor	B	157.70.109.241			
	internationnewsmedia	B	157.69.141.26			
	respecttomajorthings.c	B	199.250.229.83			
	HELO 07ouq.ssl-certific	S	42.171.11.34			
	EHLO senlicand.com	U	44.21.93.23			
	ho8mh.ssl-certificate3	U	42.209.133.218			
	hqqz6.ssl-certificate96	U	42.209.184.92			
	uaeive.org	B	111.254.149.208			

Table 10. Matrix A of HMM State Transition Probabilities

Transition Source\Target	Suspicious	Unknown	Benign
Suspicious	.206	.058	.058
Unknown	.058	.176	.058
Benign	.058	.058	.147

Table 11. A Part of Matrix B, of Probability of Observation O in HMM States

O\state	Suspicious	Unknown	Benign
<0,1,0>	.853	.066	.080
<0,2,1>	.611	.130	.388
<1,0,0>	.644	.172	.206

HMM observables are discrete. Clearly, the more observables an HMM has, the larger the required training set becomes. In particular, floating point values induce a potentially infinite set of observables.

To solve this problem, we introduce a quantification step, where observables that have very large ranges, such as floating point or string observables, are quantized into a small set of discrete possibilities. For example, consider a concrete event *SendingIP*, which has a huge ranges of possible values; we decided to quantize this range into four quantized values:

- Type 1 represents the beginning of the IP address starting with 63.
- Type 2 represents the beginning of the IP address starting with 157.
- Type 3 represents the beginning of the IP address starting with 45.
- Type 4 represents any IP address that is not specific within Types 1–3.

The quantization operation is executed using a Python script. Listing 1 shows one such quantization code snippet.

```
import sys

list =  sys.argv
if (len(list) != 2):
   print(“CallError: expecting two arguments (path to this script and a string of data); got
%d” %len(list))
   sys.exit(0)
#print (“%s %s” %(‘data:’,list[1]))
cells = list[1].split(“_”) #split on “_”
#print (“%s %s” %(‘cells:’,cells))

#quantization
outStr = ““
SendingIP = ““
for cell in cells:
   cell = cell.replace(,’”‘““);
   cell = cell.replace(,”‘“““);
   #****** THIS IS WHERE YOU MAKE CHANGES TO THE CODE TO REFLECT
YOUR QUANTIZATION


   if cell.startswith(‘63’):
      SendingIP=“TYPE1”
   elif cell.startswith(‘157’):
      SendingIP=“TYPE2”
   elif cell.startswith(‘45’):
      SendingIP=“TYPE3”
   else: SendingIP=“TYPE4”

   print(SendingIP)
```

Listing 1. Python Code Quantization

D. GENERATING CODE FOR THE HYBRID RM MONITOR

In this phase, we generated code for the RM block of Figure 9 Chapter IV.A. This step is completely automated, using the *dtracg* tool [22], which implements the algorithm described in Chapter IV.B [22].

E. RUNTIME CSV'S

In the final phase of this thesis, we perform RM with hidden data using incoming streams of emails represented as a runtime csv file (aka runtime table) depicted in Table 12. Our typical runtime table has four columns: date, sending host, and sending IP. Clearly, the hidden state column is not presented in the runtime table—it is now using the HMM, as explained in Chapter IV.B.

As explained in Chapter IV.B, the hybrid RM method uses the runtime-table and the outputs of the HMM Alpha method as its inputs. Hence, we first executed the Alpha method using the HMM and the runtime table. This step is automatic, using the *dtraalpha* tool [22].

Table 12. Runtime CSV File

Date	Sendinghost	SendingIP
2014-09-08 13:59 UTC	3ff7b9e2.cst.lightpath.net	63.247.185.226
2014-09-08 15:59 UTC	3ff7b9e2.cst.lightpath.net	63.247.185.226
2014-09-08 16:35 UTC	3ff7b9e2.cst.lightpath.net	63.247.185.226
2014-09-08 17:22 UTC	3ff7b9e2.cst.lightpath.net	63.247.185.226
2014-09-08 17:29 UTC	prisma-lan-64.68.213.1.borde	64.68.213.1
2014-09-08 20:21 UTC	host064170.metrored.net.mx	201.130.71.170
2014-09-09 04:07 UTC	unknown.telstraglobal.net	202.126.172.110
2014-09-09 13:16 UTC	3ff7b9e2.cst.lightpath.net	63.247.185.226
2014-09-09 13:52 UTC	3ff7b9e2.cst.lightpath.net	63.247.185.226
2014-09-09 14:04 UTC	3ff7b9e2.cst.lightpath.net	63.247.185.226
2014-09-09 15:38 UTC	US, Houston - MCI Communica	63.124.7.24
2014-09-09 18:26 UTC	mail.strataproducts.com	209.156.34.194
2015-09-23 20:01 UTC	host1.west-sands.com	49.231.227.9
2015-09-27 06:58 UTC	mta1234.mail.bf1.yahoo.com	157.11.65.180
2015-10-19 13:57 UTC	readytobepartofanything.com	157.69.181.175
2015-10-19 14:09 UTC	realwindowtestingy.com	45.57.234.181
2015-10-19 15:35 UTC	mirtelecom-bd.net	14.5.18.204
2015-10-20 19:29 UTC	realwindowtestingy.com	45.57.200.150
2015-11-03 23:31:02 UT	imortexport67.com	157.11.98.183
2015-11-04 10:11 UTC	specific.abidening.com	23.238.14.169
2015-11-04 10:36:42 UT	onesuccessfulltransport.com	157.70.109.241
2015-11-04 12:36:57 UT	internationnewsmediaworks.c	157.69.141.26
2015-11-04 13:00:42 UT	respecttomajorthings.com	199.250.229.83
2015-11-27 14:25:37 UT	HELO 07ouq.ssl-certificate342	42.171.11.34

F. HYBRID RUNTIME MONITORING EXAMPLE

Hybrid RM is where the rubber meets the road, as far as this thesis is concerned. We executed the hybrid monitor (the output of the *dtracg* tool discussed in section V.D), using yet another tool: *dtrarm* tool [22]. The results of this step are in Listing 2, which shows a list of probabilities associated with the instance of Rule 9 depicted in Figure 8. Note that the probability listed in row *i* is the probability this rule instance reached a Flag state. For example, row 1 through 13 shows 0% probability of reaching the Flag state; on row 21, that probability reached 73%. Indeed, on row 21, *SendingIP* being unknown (with probability 92%) for the second time within 30 minutes induces the probability of Flag to jump to 73% given that the rule's NL is "*Flag whenever some pair of consecutive emails whose SendingIP is unknown is less than 30 minutes apart.*"

```
OK! The following is a list of probability values, one per cycle (CSV file row), being the
probability of the monitor reaching the Flag state in that cycle
Row 1: probability of Flag=0.0
Row 2: probability of Flag=0.0
Row 3: probability of Flag=0.0
Row 4: probability of Flag=0.0
Row 5: probability of Flag=0.0
Row 6: probability of Flag=0.0
Row 7: probability of Flag=0.0
Row 8: probability of Flag=0.0
Row 9: probability of Flag=0.0
Row 10: probability of Flag=0.0
Row 11: probability of Flag=0.0
Row 12: probability of Flag=0.0
Row 13: probability of Flag=0.0
Row 14: probability of Flag=0.0
Row 15: probability of Flag=0.0
Row 16: probability of Flag=0.0
Row 17: probability of Flag=0.0
Row 18: probability of Flag=0.0
Row 19: probability of Flag=1.1102230246251565E-16
Row 20: probability of Flag=1.1102230246251565E-16
Row 21: probability of Flag=0.7312539202828306
Row 22: probability of Flag=0.7312539202828306
Row 23: probability of Flag=0.7312539202828308
Row 24: probability of Flag=0.7312539202828305
Done
```

Listing 2. Probability Values, One Per Cycle, of the Monitor Reaching the Flag State in Each Cycle (CSV File Row).

THIS PAGE INTENTIONALLY LEFT BLANK

VI. CONCLUSION AND FUTURE RESEARCH

Malicious emails continue to cause a significant challenge because of the threat that they present. Measures that have been imposed to help in dealing with the malicious have not been successful. Potential threats imposed by the malicious emails adjust to the inventions that are introduced. Even though the complete eradication of programs that are malicious appears to be a difficult task, the information possessed regarding the availability of the malicious programs is crucial in limiting the threat that exists.

In this thesis, we have exhibited a technique to perform RM with hidden data. The motivation behind this thesis is to determine whether this technique can be used for the detection of malicious emails. The high-level strategy for identifying such malicious emails is to monitor the sequences and temporal pattern behavior.

An additional property of out technique is its capability to handle datasets where not all data is observable. The abovementioned time and sequencing monitoring capabilities allows us to reveal potentially malicious email by not only using individual emails events, but sequences of such.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] R. Sedgewick and K. Wayne, *Algorithms*, 4th ed. Boston: Addison-Wesley Educational Publishers, 2011. pp. 30–35.

[2] D. Drusinsky, “Runtime monitoring and verification of systems with hidden information,” *Innovations in Systems and Software Engineering*, vol. 10, no. 2, pp. 123–136, 2014. Available: http://www.time-rover.com/articles.html

[3] A. M. Fiskiran and R. B. Lee, “Runtime execution monitoring (REM) to detect and prevent malicious code execution,” Princeton University. *ICCD 2004, IEEE International Symposium*, pp. 452–457, October 2004. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1347961

[4] A. A. Slack, “Digital authentication for official bulk email,” M.S. thesis, Dept. of Comp. Eng., Naval Postgraduate School, pp. 5–10, 2009. Available: http://cisr.nps.edu/downloads/theses/09thesis_slack.pdf

[5] E. Sharf, “Fake malware notifications from “Websense Labs,” Websense Security Labs Blog, 2011. Available: https://blogs.forcepoint.com/security-labs/fake-malware-notifications-websense-labs Accessed August 2016.

[6] J. W. Ragucci, S. A. Robila, “Societal aspects of phishing,” *Technology and Society, 2006. ISTAS 2006, IEEE International Symposium*, pp. 1–5, June 2006. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4375893. Accessed September 2015.

[7] B. Brewin, “DOD battles spear phishing,” The Business of Federal Technology, 2006. Available: https://fcw.com/articles/2006/12/26/dod-battles-spear-phishing.aspx. Accessed August 2016.

[8] D. Drusinsky. “Rules for business.” Rules4Business. Available: http://www.rules4business.com/acmeBank/index.html

[9] D. Drusinsky, “UML-based specification, validation, and log-file based verification of the Orion Pad Abort software,” technical report NPS-CS-10-007, Naval Postgraduate School, pp. 1–24, 2010. Available: http://calhoun.nps.edu/bitstream/handle/10945/549/NPS-CS-10-007.pdf

[10] D. Drusinsky, *The Temporal Rover and the ATG Rover*. Springer-Verlag Lecture Notes in Computer Science, 1885, pp. 323–329.

[11] Havelund, K., Rosu, G., “An Overview of the Runtime Verification Tool Java PathExplorer,” *Formal Methods in System Design*, vol. 24, 189–215, 2004.

[12] U. Sammapun, I. Lee, and O. Sokolsky, "RT-MaC: Runtime Monitoring and Checking of Quantitative and Probabilistic Properties," *Proc. 11th IEEE Int'l Conf. Embedded and Real-Time Computing Systems and Applications*, IEEE, pp. 147–153, 2005.

[13] The StateRover. (2016, June 10). *Time-Rover*. [Online]. Available: http://www.time-rover.com. Accessed June 10, 2016.

[14] P. Meseguer and A. D. Preece, "Verification and validation of knowledge-based systems with formal specifications," University of Aberdeen, pp. 1–4, 1990. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.17.7692&rep=rep1&type=pdf

[15] D. Drusinsky, J. B. Michael, and M. T. Shing, "A visual tradeoff space for formal verification and validation techniques," *Systems Journal, IEEE*, vol. 2, no. 4, pp. 513–519, Dec. 2008.

[16] K. Shimizu, D. L. Dill, and A. J. Hu, "Monitor-based formal specification of PCI," *Formal Methods in Computer-Aided Design*, vol. 1954, pp. 372–390, Jun. 2000.

[17] D. Drusinsky, J. B. Michael, T. W. Otani, and M. T. Shing, "Validating UML statechart-based assertions libraries for improved reliability and assurance," in *SSIRI'08 Second International Conference*, Yokohama, Japan, pp. 47–51, 2008.

[18] J. J. Galinski, "Formal Specifications for an Electrical Power Grid System Stability and Reliability," M.S. thesis, Naval Postgraduate School, pp. 1–11, 2015. Available: http://cisr.nps.edu/downloads/theses/15thesis_galinski.pdf

[19] Rabiner, L.W., "A tutorial on hidden Markov models and selected applications in speech recognition," *Proc. of the IEEE*, vol. 77, no. 2, 1989.

[20] Hidden Markov model. (2016, June 10). *Wikipedia*. Available: https://en.wikipedia.org/wiki/Hidden_Markov_model. Accessed June 10, 2016.

[21] D. Drusinsky, "Behavioral and temporal pattern detection within financial data with hidden information," *J. UCS*, vol. 18, no. 14, pp. 1950–1966, Jul. 2012.

[22] D. Drusinsky, "A hidden Markov Model based runtime monitoring tool," technical report NPS-CS-16-001, Naval Postgraduate School, pp. 1–34, 2016. Available: http://calhoun.nps.edu/handle/10945/47575

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California